Lab 3 Thursday 1 - 4

Lab Introduction

Hello, coding champs!

You just got assigned to your first job as a Game Developer, You are given several header files and are required to implement and **test** your code

The company wants you to make sure you **validate your inputs** as previous bugs have caused severe losses for the company on previous projects.

You will create snake game in which a snake that moves around a grid, and can pick and remove from coins its inventory.

Let us start describing the classes

Classes

1 Coin

```
1   class Coin {
2   private:
3       int value;
4
5   public:
6       Coin(int value) {this-> value = value;};
7       void printInfo();
};
8
9
```

This class only has a constructor and a `printInfo()` function.

1.1 Constructor

Sets the information, make sure to apply the needed validation checks.

1.2 printInfo()

Prints the following "`10 value`"

2 Pacman

```
class Pacman {
1    private:
2        //the pacman should contain
3        an array of coins how can we
4        define it? and what is al
5        relation between them
6        //how to define the constant
7        MAX_COINS
8        int coinCount;
9        int totalValue;
10   public:
         ~Pacman();
         void printInfo();
         bool addCoin(Coin * coin);
         bool removeCoin(int index);
};
```

2.1 Destructor

Check if any object needs to be cleaned.

2.2 printInfo

Print the number of coins in the inventory and the information of each coins and total value of coins.

```
1   Pacman has 2 Coins
2   Coin 1: 10 value
3   Coin 2: 15 value
    Total value: 25
```

2.3 addCoin

Adds a coin to the pacman and performs the required check accordingly.
Returns true if coin was added succesfully, false otherwise

2.4 removeCoin

Removes the coin from the inventory and deletes it.

Returns true if coin was removed succesfully, false otherwise

3 Grid

```
class Grid
{
private:
    //what is the relation between pacman and the grid
    // you should define a pacman here
    Coin * CoinGrid[GRID_SIZE][GRID_SIZE];
public:
    Grid();
    ~Grid();
    void setCoin(int x, int y, Coin * coin);
    Coin * popCoin(int x, int y);
    void printGrid();
    void movePacman(int x, int y);
    Pacman getPacman();
};
```

3.1 Constructor

The grid should have a 2D array of coins, all pointers should be initialized to null in the constructor.

3.2 Destructor

Do we need to delete anything if we delete the Grid class ?

3.3 printGrid

The grid should be printed as a 2D matrix, If there is a weapon in the (x,y) spot we should print W else we should print -

```
C----
--------
Coin at (0,0): 10 Value
```

This is an example for the grid that has one Coin at (0, 0)

3.4 setCoin

This adds a coin to the grid. if there is an already existing coin, the old coin is deleted and replaced by the new one.

3.5 popCoin

Removes coin from the grid and returns a coin pointer.

3.6 movePacman

Takes an x and y coordinate as a parameter and moves the pacman to that location. If that location contains a coin, it should be automatically picked if possible.

3.7 getPacman

Gets player object

Important Notes

You'll be required to test the classes individually before constructing your main function.

I would advise you to go with the following order
Coin ⟶ Test Coin ⟶ Pacman ⟶ Test Pacman ⟶ Grid ⟶ Test Grid

Testing

To test our implementation, we will create different functions inside our main file, each function will test one of the classes.

```
// main.cpp

void testCoinClass() {
    // test code
}

void testPacmanClass() {
    // test code
}

int main() {
    testCoinClass();
    testPacmanClass();
    return 0;
}
```

Testing Coin

Testing the coin class will consist of creating two coins.

- Coin 1: 10 value
- Coin 2: 20 value

We will then print the Coins, that are succesfully created.
If the coin creation fails, print an error message of your choice.

Testing Pacman

Scenarios

We should print pacman info between each step just like in Pacman 1 even if not mentioned.

Print any error message if the step is invalid

1. Pacman 1
    - Create Pacman
    - Print Pacman Info
    - Add 1 coin
    - Print Pacman Info
    - Add 2 more coins
    - Print Pacman Info
2. Pacman 2
    - Create Pacman
    - Remove coin at index 2
    - Add a new coin
    - Remove coin at index 2
    - Remove coin at index 1
3. Pacman 3
    - Create Player
    - Add 5 coins to Pacman
    - Add 1 coin to Pacman
    - Remove Coin at index 3

Testing Grid

Scenarios

1. Grid 1
    - Create Grid (would you need to create something before creating the grid ?)
    - Add Coin at (7, 4)
    - Add Coin at (1, 3)
      (Coins should appear in the first row and third column)
    - Add another coin with different properties at (1, 3)
    - Move pacman to (7, 4)
    - Move pacman to (1, 3)
2. Grid 2
    - Create a grid
    - Add 6 different coins on the grid
    - Make the pacman move to these 6 locations